Australian Government
Department of Defence
Defence Science and
Technology Group

Auxiliary Library Explorer (ALEX) Development

Alex Bunting[1], *Stephen G. M^cAteer*[2] *and Justin Beck*[2]

[1] **University of New South Wales**
[2] **Joint and Operations Analysis Division**
Defence Science and Technology Group

DST-Group–TN–1492

ABSTRACT

In this report we describe the development of the *Auxiliary Library Explorer* (*ALEX*) which was carried out as part of a DST Group summer vacation scholarship in 2013–14 by Alex Bunting. *ALEX* is a prototype tool motivated by *Google Scholar* which uses the DST Group for-official-use-only (FOUO) internal report database as its input. It provides DST Group researchers with an additional discovery capability and is available (at the time of publication) on the DST Group internal website. We use *ALEX*'s underlying database to explore DST Group's collaboration network and discuss a preliminary result supporting the assertion that inter-divisional collaboration may occur within sites more than between sites.

RELEASE LIMITATION

Approved for Public Release

Published by

Joint and Operations Analysis Division
Defence Science and Technology Group
506 Lorimer St,
Fishermans Bend, Victoria 3207, Australia

Telephone: *1300 333 362*
Facsimile: *(03) 9626 7999*

AR-016-505
February, 2016

APPROVED FOR PUBLIC RELEASE

Auxiliary Library Explorer (ALEX) Development

Executive Summary

In this report we describe the work completed by DST Group summer vacation scholar, Alex Bunting in the summer of 2013–14. The project uses big data techniques to develop a *Google-Scholar*-like knowledge discovery capability based on the DST Group internal report database. The major output of the scholarship is the *Auxiliary Library Explorer* (*ALEX*) which is, at the time of writing, available to DST Group staff via the library intranet site.

ALEX uses data from DST Group's Research Services Group (RSG) repository on the Protected Research Network, the classification of which is up to for official use only (FOUO). It provides value to DST Group by prototyping a more intuitive and functional interface than the standard front-end to the *DSpace* database currently in use on the library site. It was built using free, open-source software (FOSS) tools.

In constructing *ALEX*, the techniques we used include,

- extract, transform and load (ETL)
- optical character recognition
- data schema development
- Structured Query Language (SQL) database development
- data mining
- interface development
- data visualisation and analysis.

These techniques are common to a number of problems addressed by DST Group and added insight to how they could be used more broadly, particularly as DST Group moves to embrace big data technology.

We also use *ALEX*'s underlying database to explore DST Group's collaboration network. The collaboration network is generated by report co-authorship — it is a graph where the nodes are authors and a pair of nodes is connected with an edge when the authors represented by the nodes collaborate on a DST Group report. We present a preliminary result supporting the assertion that inter-divisional collaboration may occur within sites more than between sites.

THIS PAGE IS INTENTIONALLY BLANK

Contents

Figures

Glossary

AJAX	Asynchronous Javascript and XML
ALEX	Auxiliary Library Explorer
AOD	Air Operations Division
AVD	Air Vehicles Division
CSS	Cascading Style Sheet
CSV	comma separated values
DST Group	Defence Science and Technology Group
ETL	extract, transform and load
EWRD	Electronic Warfare and Radar Division
FAQ	frequently asked questions
FOUO	for official use only
FOSS	free, open-source software
HPPD	Human Protection and Performance Division
HTML	Hypertext Markup Language
ISRD	Intelligence, Surveillance and Reconnaissance Division
ITD	Information Technology Division
JOAD	Joint and Operations Analysis Division
JOD	Joint Operations Division
LOD	Land Operations Division
MOD	Maritime Operations Division
MPD	Maritime Platforms Division
OCR	optical character recognition
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor (recursive acronym)
POI	Poor Obfuscation Implementation
RAM	random access memory
RSG	Research Services Group
SQL	Structured Query Language
TIFF	Tagged Image File Format
TF-IDF	Term Frequency–Inverse Document Frequency

UCS	Unified Computing System
UTF-8	UCS Transformation Format 8-bit
VBA	Visual Basic for Applications
WSD	Weapons Systems Division
WRE	Weapons Research Establishment
WSRL	Weapons Systems Research Laboratory
XML	Extensible Markup Language

1 Introduction

The *Fleet Data* initiative within the Joint and Operations Analysis Division (JOAD) aims to modernise the Royal Australian Navy's (RAN's) approach to data collection and storage. The *big data* philosophy is to collect all data and store it in such a way that it can be processed using massively parallelised algorithms. This is sometimes characterised as the "$N = all$" approach.

In 2013, *Fleet Data* proposed a summer vacation scholarship with the aim of using *big data* tools and techniques to create an internal website similar to *Google Scholar* for DST Group's publications classified up to for-official-use-only (FOUO)[1]. Figure 1 is a screenshot of the final product: *Auxiliary Library Explorer (ALEX)* [1]. *ALEX* prototypes a report library interface that would allow DST Group staff to explore the report database in ways unavailable under *DSpace*, the current report library interface.

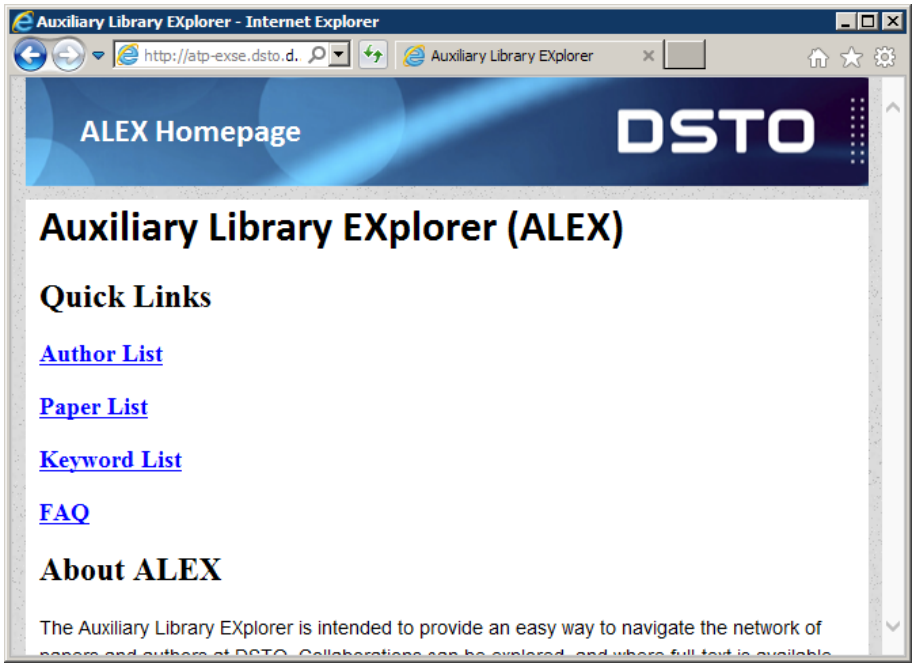

Figure 1: A screenshot from the ALEX *homepage.*

The summer vacation scholarship program runs for approximately 14 weeks and involves up to 70 students across DST Group. Alex Bunting[2], was selected for this project. Alex conducted nearly all the work towards the completion of the project.

Prior to the student's arrival, a database of all DST Group reports held on the Protected Research Network was provided by DST Group's Research Services Group (RSG). This database notionally contains all reports DST Group has produced at or below the FOUO

[1] This corresponds to RESTRICTED under the former classification system.

[2] At the time of the project, Alex had completed the third year of a four year double degree in computer and mechanical engineering at The University of New South Wales.

classification. However, there are known to be gaps, especially in older works which have not all been digitised. The period of time covered by this database is 1963–2013.

In this report we provide an account of the work undertaken by the student to develop *ALEX*. At the time of writing, *ALEX* is available from RSG internal website and was advertised in the DST Group internal news. Website analytics was used to establish that hundreds of DST Group personnel had browsed *ALEX* pages, and we have received some positive feedback from users.

We also use *ALEX*'s underlying database to explore DST Group's collaboration network. The collaboration network is generated by report co-authorship — it is a graph where the nodes are authors and a pair of nodes is connected with an edge when the authors represented by the nodes collaborate on a DST Group report. We present a preliminary result supporting the assertion that inter-divisional collaboration may occur within sites more than between sites.

In Section 2, we discuss the preparation of the data for analysis. In Section 3, we describe the structure of the *ALEX* database. In Section 4, we describe the algorithms used to populate the *ALEX* database. The final product of *ALEX* was a web front-end. In Section 5, we describe how this front-end was generated. Some descriptive analysis of the structure of DST Group's collaboration network was carried out, and the process of visualising the data is described in Section 6. In Section 8, we describe work that could follow on from this project.

The source code and data referred to in this report has been placed on Objective (object id `fAV1010092`, database `ednrdm20`). Appendix A describes the process of setting up *ALEX* from scratch using this source code and data files.

2 Data preparation

The vast bulk of effort dedicated to big data lies in the process known as extract, transform and load (ETL) [7] — raw, possibly unstructured data is cleaned and reshaped in preparation for analysis. In this section, we describe the data sources and the ETL process we applied to them.

The RSG provided three important sources of information relating to DST Group reports from the period 1963–2013. These sources are: a Structured Query Language (SQL) file called `outfile` (no extension), a metadata file `DSpacemetadata.csv`, and an archive file `assetstore-20131112.tar.gz`.

PostgreSQL is used to read the SQL file `outfile`, which reconstructs the *DSpace* database on a local SQL server. This database does not include metadata or the full-text of the DST Group reports. We look into the contents of the database further in the following section. The database *does* contain the metadata about each paper, including title, author names, abstract and DST Group identifier.

The full-text of the data dump is contained within the `assetstore-20131112.tar.gz` file. This file needs to be unzipped twice (using *7-Zip* or other software capable of extracting `.gz` and `.tar` files). This file contains many nested folders each named using a pair decimal digits. Within the third level of these nested folders there are one or two files,

also named using decimal digits: the first six digits correspond to the names of the folders containing the files. An example of this is shown in Figure 2.

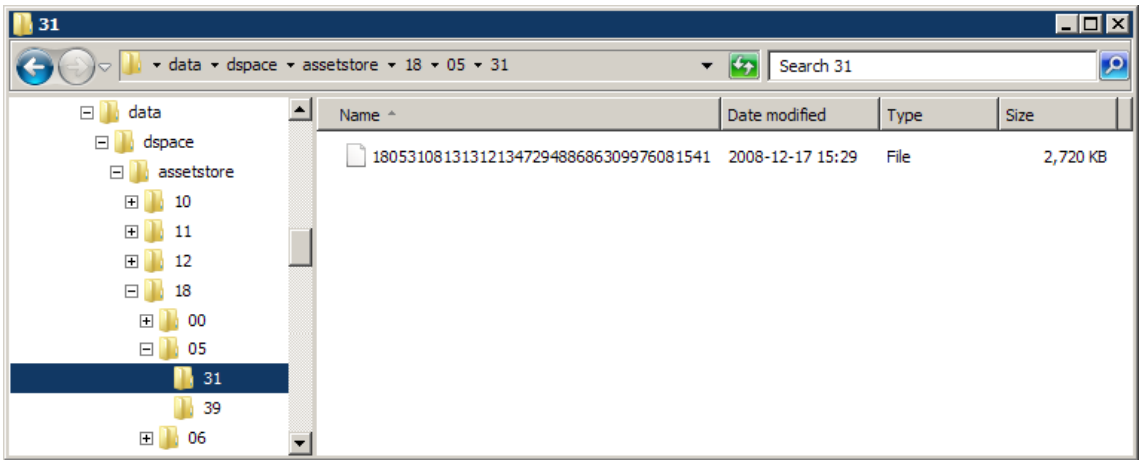

Figure 2: An example of the long file names and their relationship to the directory structure. In this case we can see that the first six digits of the filename (`180531`*) correspond to the directory structure (*`.../18/05/31/`*).*

2.1 Metadata extraction

The information contained within the *DSpace* database is not in a useful format for the purposes of constructing *ALEX*. The schema of a *DSpace* database is very general and therefore complex; Figure 3 is a graphical representation of this schema. The full size image is available on the *DSpace* development website [3]. Note that not all tables in this schema are used by the RSG reports database.

For the purpose of creating *ALEX*, many of the 44 or so tables within the full schema are discarded. These include information such as the identity of the file uploader, and various checksum and registration data — useful information for the library database manager but less interesting for exploring the library content. Instead, to get useful information about the library content, we need to combine the metadata about the papers with the information in the *DSpace* database about the files of the full-text (the files depicted in Figure 2).

2.1.1 Reading the metadata

The metadata file is originally in a format difficult to work with. The primary difficulty is that there are many columns either empty, containing unnecessary information, or duplicate columns. These duplicate columns are apparently the result of the data being in different languages — `English`, `Australian English`, `American English`, and `unspecified`. Many papers have their subjects listed (and in some cases duplicated) across the four language columns. In order to access these subjects more easily, a consolidation across these columns is performed.

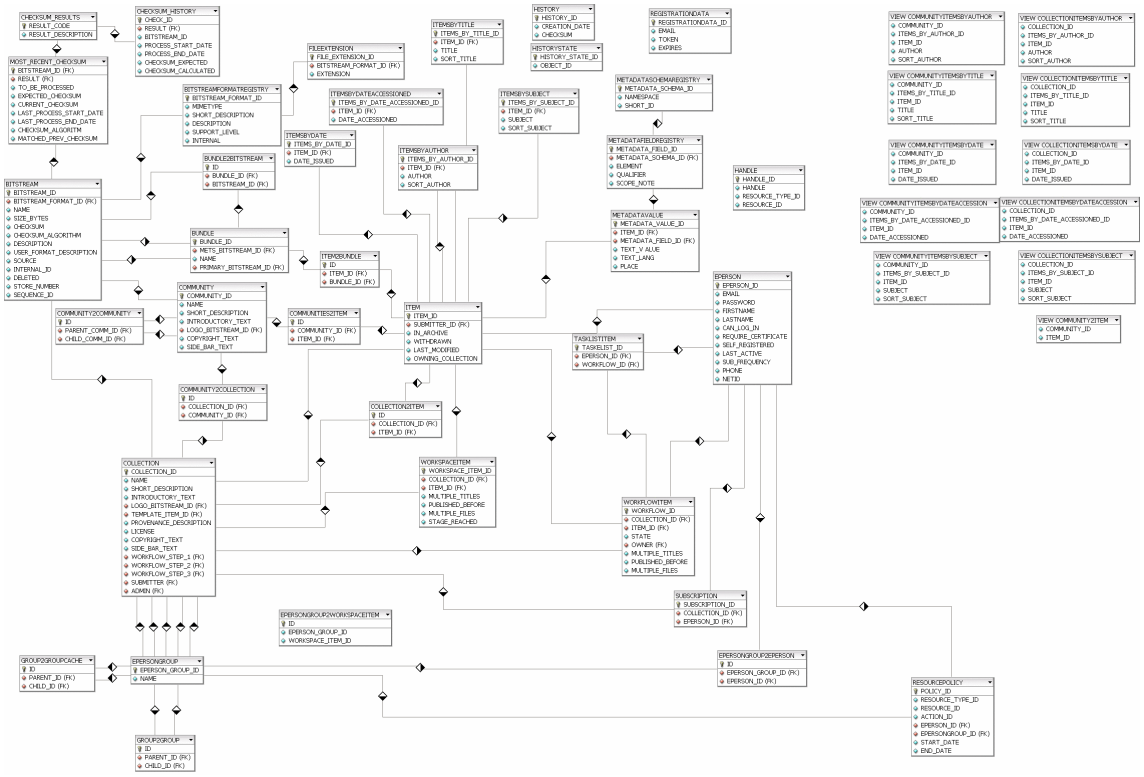

Figure 3: A graphical representation of the DSpace database schema. This schema is a very general and therefore complex [3].

To begin with, the duplicate language columns are deleted after concatenating their content into a single column using *Excel* expressions to copy contents of non-empty cells. This is a laborious manual task and could probably have been avoided by using *Java* code to read the data directly from *Excel*. In fact, it might be even easier to leave the data as a comma separated variables (CSV) file and read the data in with *Java*, although this could create other issues with parsing the CSV files when content includes longer strings with commas. As implemented, the tidied metadata is saved as an *Excel* spreadsheet, a `.xlsx` file, which had to be accessed using the *Apache Poor Obfuscation Implementation (POI)* library [5].

The *Apache POI* library is used to interact with *MS Office* applications. However, as free software still under development, the documentation is difficult to read and the dependencies it requires are not obvious at the outset. To use *Apache POI* with *Excel* files, all `.jar` files within the unzipped folder and all `.jar` files within the `lib` and `ooxml-lib` files need to be included in an *Eclipse* project build path. Data formatted as a spreadsheet can then be accessed using a workbook object which provides methods for accessing sheets, rows and cells.

2.1.2 Linking with the database tables

To be able to link the metadata to their full-text it is necessary to integrate the metadata with the relevant parts of the *DSpace* database. The tables of interest for this purpose are as follows.

`Item`	List of items contained within the metadata file. ID numbers in this table correspond to ID numbers in the metadata and provide the link between metadata and physical file.
`Item2bundle`	A link table that converts item IDs to bundle IDs.
`bundle2bitstream`	Another link table that converts bundle IDs to bitstream IDs.
`bitstream`	A very important table that lists the internal identifier of all the bitstream files within the library asset store. These internal IDs are the names of the files, where the first six digits can be used to find the location of the file in the asset store. It also includes the original file name of the file before they are given the internal ID for storage by the database software.
`bitstreamformatregistry`	A lookup table that links the formats listed in the `bitstream` table with the file extension of each file.
`fileextension`	Lists the file extension and file type name of all the file types in the database.

A complex SQL query is used to join all the above tables.

This is first implemented using the `MakeFullDatabaseapp` *Java* project, which performs an SQL query on the *DSpace* data to return a list of items for which the original file name is not a blank page, license or some other uninteresting item. This criterion is applied in order to filter out which papers actually have full-text available, for the purpose of performing text analysis. The metadata for that particular item is then looked up by item number and the entry is added to a new SQL table — `librarysummary`.

Metadata is extracted for the remaining papers — those without full-text. This is performed with the `AddPapers` project. Here the project approaches the problem from the other direction — starting with the metadata rather than the SQL database. However since it is already known that all the papers being added have no full-text in the library, there is no integration with the existing *DSpace* information. Therefore the new metadata is dumped straight into `librarysummary`.

To properly reconstruct the `librarysummary` with a new library data dump would require a combination of the two projects. Ideally the method of `AddPapers` (step through every entry in the metadata) would be combined with the method of `MakeFullDatabaseapp` (look up every item to see if it has full-text according to *DSpace*). This could be accomplished with small modification of `AddPapers`. To make this process faster, this project could be constructed to read multiple columns of the same language and take the first non-empty one, thus avoiding the tedious metadata tidying described in Subsubsection 2.1.1.

2.2 Full text extraction

With the `librarysummary` table populated, the next step is to extract a library of text-only versions of all available full-text files from the asset store. Of the 16,000 or so papers listed in the metadata, only around 5,000 have full-text available. Predominantly the asset store includes searchable Portable Document Format (PDF) files, with embedded text that can be extracted to a text file.

When the text extraction was initially performed (on the *embedded* PDF text) the output was deceptive. All PDFs extracted without warnings or errors, but on inspection the output text files were filed with garbled text (and less of it than would be expected in a full paper). This results from scans of older documents, where some text had been recognised by OCR and added on the front page by the scanner, but all remaining text had been ignored.

2.2.1 *MS Word* documents

The library also includes approximately 200 word documents which need to be extracted to plain text files. These are extracted by opening each document and saving it as a text file using a Visual Basic for Applications (VBA) script. The method of execution is not straightforward — it requires a *Java* routine to move all the *MS Word* documents to a directory in which the VBA script is called. The VBA script then steps through all the files which require saving as text. In a future implementation, a better solution would be for the presence of a *MS Word* document to be detected during text extraction operation and a VBA script or *Windows PowerShell* script could be called to save it as text, rather than having to do a separate operation.

2.2.2 PDFs with text data

PdfToText [4] (part of the *XPDF* package) is a free command-line PDF text extraction utility for *MS Windows* which is used to extract plain text from PDFs that are 'searchable' (that is, which already had embedded text). It has no effect on documents without embedded text, however all DST Group report scans had at least some text recognised with optical character recognition (OCR) embedded in them during the scanning process. The quality of this OCR is very low: missing most words, not recognising two-column format, and so on. The final output is largely unreadable (even by manual human inspection) and leaves little hope for automated analysis.

The text extraction is implemented in the project `PDFTextExtraction03pdftotextapp`.

The UTF-8[3] text format is not used in the extraction process; however, this would be preferable since the OCR program used only works with this text format.

The *Java* OCR libraries, *PdfToText*, *PDFBox* and *iTextPDF*, were all considered for use in this project. *PdfToText* was selected because it generates the best results; specifically, it is better at dealing with two-column formatting and recognition of ligatures.

2.2.3 PDFs without text data

There is no obvious way from the metadata or the PDF itself to tell if the text embedded in a PDF is of a high standard. The metric that is used in this project to determine whether a PDF needs to have optical character recognition applied to it is a simple word-count on the text output from running PDF extraction (described above). We observed that documents without proper embedded text result in many artefacts such as non-alpha-numeric symbols and single-letter words; these are ignored in the word-counts. The number of remaining words in each document's embedded text is then counted, and if this count is less than 750 (which would indicate a very short report if it were truly the paper's full-text), OCR is applied to the document.

The OCR is performed by *Tesseract OCR* [10], a free OCR program initially developed by Hewlett-Packard and then further developed by Google. *Tesseract OCR* only operates on very specific types of Tagged Image File Format (TIFF) files, so it is necessary to convert the PDF to a series of images. This is achieved using *Imagemagick* [6] — a FOSS suite of command-line image manipulation tools with extensions to allow use in many major programming languages. In order to read the PDF, *Imagemagick* uses *Ghostscript*, a free open source PDF viewer.

For each PDF, the number of pages is extracted using *PDFBox*, and then each page is converted to a TIFF file and analysed using *Tesseract OCR* individually.

Tesseract OCR takes as input, a TIFF file or a number of TIFF files and outputs UTF-8 encoded text files. By inspection, *Tesseract OCR*'s output is of a high standard as long as the scan is of a reasonable quality, and it has nearly flawless output on more recent scans.

This process's most recent implementation is the project `TesseractOCRv04` which suppresses an error caused by jagged lines in old PDF scans and analyses the pages individually as discussed above.

At the end of this process, all of the embedded full-text and extracted full-text locations are linked in to the database to facilitate automatic access.

3 Database structure

In the previous section, we described the process of transforming the metadata so that it is better suited to the purposes of this study. In this section we describe the underlying structure of the database used in *ALEX*.

[3] Unified Computing System (UCS) Transformation Format 8-bit

The *ALEX* database has a graph structure, with authors, papers and topics forming the *nodes* and their interrelations forming *edges* (that is, connections between the nodes).

3.1 Node tables

Three node tables list all the important items in the database. These node tables are represented schematically as the circles in Figure 4. The three node tables in the *ALEX* database are described below.

`librarysummary` This table lists all the papers in more useful form than the original *DSpace* metadata and also includes links to the full-text original and extracted-text-only file locations.

`authorsummary` This table is a list of all authors. It could be expanded in the future to include other interesting metadata such as years of active publication, divisions operated in, and so on.

`keywordsummary` This table gives a list of all the keywords (that is, the topics covered by each paper). These keywords are split up into those which were manually entered into the papers' metadata during the publication process, and those automatically extracted. The automatically extracted keywords are indicated by an asterisk at the start of the keyword. For example, `radar` is a manually entered keyword, whereas `*radar` has been automatically extracted.

3.2 Relation tables

There are six tables which create the edges linking the nodes. There is one table for every type of link between node types (including nodes linking to themselves), some of which are undirected while other links are directional. These link tables are represented with arrows in Figure 4 and described below.

`author2paper` This table captures authorship. It lists the links between authors and the papers they have written as extracted from the metadata.

`author2author`	This table captures collaboration. It lists the links between authors based on them having co-authored a paper together. Links are uniquely identified by each paper they collaborated on. This makes it possible to work out a strength of connection based upon the number of collaborations between a pair of authors. As this link is undirected, `author1` is always chosen to be the one with a lower `author_id` in the `authorsummary` table.
`paper2paper`	This table captures which DST Group reports cite each other. It lists the links between papers through citations made in a report. These links are directed, with each link having a *citing* and a *cited* paper.
`key2paper`	This table lists the links between papers and their topics. These links are based on both manually entered metadata and automatically extracted keywords.
`author2key`	This table connects authors to keywords associated with them. It is populated using the keywords associated with their publications.
`key2key`	This table is not implemented, but could be used to link related topics. These connections could be *learned* from the structure of the database. For example, keywords shared by a given author or paper may be considered as related.

An end user of *ALEX* could use the graph structure described above to explore the report database in ways not previously available.

4 Populating the relation tables

In Section 2, we described the process of massaging the database into a more easily interrogable format (including extraction of full-text where necessary). Once this is done, we are in a position to populate the tables linking the authors, papers and keyword lists described in the previous section. In this section we describe the process of populating these tables, and detail the practical difficulties we encountered.

4.1 Author list and collaboration table

Although populating the author list and the authorship link table seems trivial from the metadata, there are complications with identifying people by name. With some exceptions, author names appear in the metadata as initials and last name only. There is therefore the possibility of authors with the same first initial and last name being erroneously

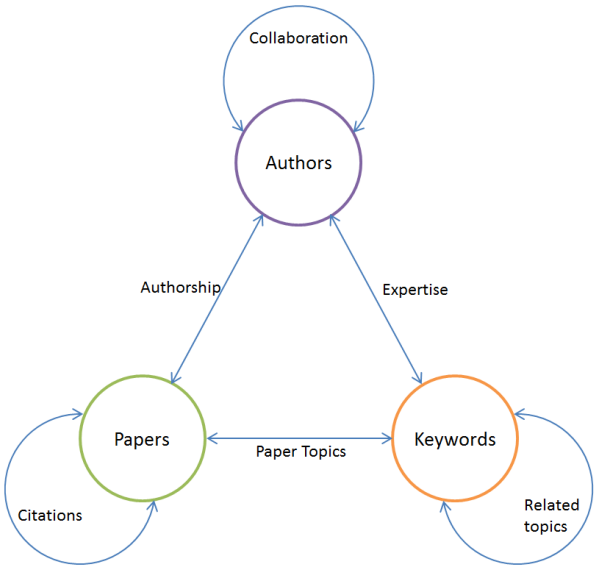

Figure 4: A diagrammatic representation of the relationship between the concepts that ALEX examines within the library data.

merged in the absence of other distinguishing features. On the other hand, authors often vary whether they include their second or third initials, or change their name entirely; this results in duplicate entries for some authors. This can be very difficult to detect automatically.

No attempt is made to overcome the issue of erroneously merged or split authors. Even working out these issues by manual inspection would be fraught with difficulty — for example if a single name appeared in two divisions, we might be inclined to split it in to two authors; but authors *do* change divisions.

With regard to the issue of the same author appearing under totally different names (due to marriage for example), automatically detecting change of last-names is nigh impossible and we did not attempt to account for this.

We took the following conservative approach to merging authors. If an author is found in the list with one initial, and one other author is found with the same surname and first initial, but additional initials, then they are merged.

We know that this metric missed some names that should have been merged — `Moon, T.` and `Moon, T.T.` are both the same author, but remained distinct because of the existence of `Moon, B.A.M.` and `Moon, J.` The advantage of this rule is that names such as `Anderson, K.` remain unmerged with `Anderson, K.L.`, `Anderson, K.W.` and `Anderson, K.R.`; there is no good way to choose which merger should happen. This is an area of possible future work with one possibility being that user feedback could be a mechanism for identifying required mergers and splits.

The final implementation of the author list generation is `AuthorListv04`. This project

extracts all the authors from the paper metadata, associates them with their respective papers, merges authors using the metric above, then populates the `author2paper` link table on the basis of the merged author names. With the `author2paper` link table populated, it is a trivial matter to step through this table for each paper and create an undirected link between all authors that wrote each paper to form the `author2author` table. For n authors on a single paper the number of links created is $\sum_{i=0}^{n-1} i = (n-1)n/2$. Note that the links between authors are non-unique, two authors that have written multiple papers together have multiple links between each other, one for each paper. This allows for a metric of collaboration strength based on the number of papers a pair of authors have published together.

4.2 Keywords

Keyword extraction is performed using the *Maui Indexer* [8] tool, which is free, open source software (FOSS). *Maui* is an implementation of the *term frequency–inverse document frequency* (TF-IDF) method, with the *inverse document frequency* based on a separate training dataset. In this implementation, phrases at the beginning and end of the document are given additional weight — the idea here is that these phrases come from the introduction and conclusion of the article in question.

Maui is run using the *Maui* example code provided without any modification — we even go to the length of placing the training dataset into the example training folder and took the output from the example output. The training dataset is made up of all papers which had both full-text and non-blank keywords from the metadata. These keywords are split up and saved in separate files (these keyword files have the same name as the original text file, but are concatenated with the extension `.key` — the format required by *Maui* for training). Running *Maui* generates sets of `.key` documents for all the target papers and places them in the output folder; this process takes several minutes. The keyword extraction is performed using the `MauiPrep` project to move text and key files to the target directories. These are the target directories specified in the `Maui1.2` example project. `Maui1.2` is then executed to perform the extraction. The extracted keywords are then inserted back into the `librarysummary` table using `AddTopicsv01` project. Keywords from papers with extracted keywords are prefixed with an asterisk (`*`) while those from the original library metadata are not modified. Finally the `keywordsummary`, `expertise` and `topics` link tables are populated using the project `KeywordListv03`.

This process has a number of limitations; firstly, the various steps should really be concatenated into a single project or script. Secondly, `Maui1.2` uses a large amount of random access memory (RAM) whilst executing — it needs to be run using 64bit *Java* and called with a virtual machine argument `-Xmx6200` to allow the *Java* virtual machine access to 6.2GB of RAM — about the maximum amount possible on an 8GB machine where the rest is being used by *Windows*. This RAM usage seems to imply that all the text files as well as all intermediary results are being held in memory during operation. While the large dataset should increase accuracy, a paper [8] comparing the various keyword extraction tools claims that *Maui* does not need a large training set to be effective. It may be possible to use only 100–1000 papers for training (we had 3000 with full-text and subjects available) to reduce the RAM usage. A further reduction in RAM requirements may be

achieved by applying *Maui* to a target set of 100 or so papers at a time. Another approach would be to use a distributed computing environment such as *Hadoop*.

There are a number of ways this process could be improved upon. The documentation recommends the data from Wikipedia Miner [9] to inform semantic analytics and relevance of technical terms. Unfortunately this requires a *Hadoop* cluster with at least 50GB of RAM (the size of the *English Wikipedia* dataset) which was not available for this project.

It may also be possible to have *Maui* assign a score to the keywords — Maui assigns a numerical value to determine the best keywords for the document during its 'filtering' step. These scores could be very useful in some later analysis for ranking the significance of words for particular authors or papers.

It might also be possible to extract keywords from those papers with human generated ones to compare the output.

Finally, there are some keywords created from artefacts left over from character encoding changes — long dashes, smart quotes and f-ligatures all being a problem. These could be remedied by applying some sort of dictionary-based spell check to the full-text documents. There are also some cases of keywords being chosen which are quite similar to each other with varying degrees of specificity — `fatigue`, `fatigue stress`, `fatigue stress crack`, `fatigue crack` all identified for one paper — there might be a way to concatenate these down to a smaller and more relevant set.

4.3 Citation

The citation network is created by using regular expressions to find DST Group report numbers which are of a consistent format of capital letters, dashes and numbers. Each document is scanned for text of the format `XXXX-XX-YYYY` (where each `X` is a letter and each `Y` is a number) so that references to other DST Group reports would be detected in the body text. This regular expression appeared to work, but more thorough testing may be required to confirm that no papers are being missed. The latest implementation of the citation network extraction is `CitationNetworkv02`.

We considered a more sophisticated technique (but did not implement it), which would give access to the full report database and not just those reports with known numbers. This technique would involve looking for reference styled author names using regular expressions. First the reference section would be found by looking for the section heading, and the author names would be extracted using regular expressions. The extracted names could then be checked against the *ALEX* database to see if they are known DST Group authors. Once a DST Group author is identified, the list of papers by that author could be retrieved, and then a fuzzy string matching algorithm could be used to see if the title of the paper occurs in the vicinity of the author's name. This would be flexible in matching DST Group publications without report numbers as well as slight variations in title (capitalisation, pluralisation, punctuation). It would also significantly reduce the complexity of the search, albeit with the slower fuzzy string matching. Since the fuzzy string matching has time order proportional to length, this matching could also be sped up by using only a subset of the words in the paper title — the first four, and a decent match for those

could be confirmed by trying to match more words in the same place. Future work could consider this approach.

5 *ALEX* website generation

With the database of links, papers, authors and keywords created, in this section, we consider the user interface for the library explorer. The *ALEX* web front-end evolved from simple collection of interlinked Hypertext Markup Language (HTML) pages to a set of generated-on-demand pages that uses a "PHP: Hypertext Preprocessor" (PHP) script and queries a SQL database for their content. This structure makes it easy to implement a simple interactive search interface.

5.1 Generation of HTML pages

There are a number of *Java* projects for creating and updating of the large number of individual pages associated with each keyword, author and paper. These pages include hard links to all the other relevant pages. However, despite a considerable amount of effort spent on developing these projects, they are no longer required due to the implementation described below that sees the pages generated by the web server as they are required. This implementation uses PHP and is described in the following subsections. The projects for generating the static HTML files are included for completeness: `AllPagesv03`, `PaperPagesv02`, `AuthorPagesv02`, `KeywordPagesv01`, `HomePagev01` (and their earlier versions).

5.2 Search functionality and the SQL database

Although third-party search vendors are available (such as Google) that can be instructed to index a specific site, this is not available for this application due to classification. Instead, the search functionality implemented in this project is based upon SQL queries of the exact string entered into a text field on the web front-end. Specifically, it searches for a case-insensitive match for the words entered in any of the target fields: author name, paper name, keyword. In order to make the site searchable using this method, it is necessary to host the SQL database on the web server so that these queries can be performed (server-side) against the database. At the time of writing, the information is stored as a *MySQL* database hosted on the Cealexse server (located at DST Group, Eveleigh). The host is `mccmysql`, and the database can be accessed using the Bash command:

```
mysql -h mccmysql -u <user> -p <password>
```

The database contains all of the tables listed in Section 3.

As the original database is a *PostgreSQL* database it is necessary to change the format. Although very similar, the implementation details mean there are a few complications with the conversion. The function `pgAdminIII` is used to dump the database to a backup file.

The options to dump only the desired tables, and to use `CREATE TABLE` and `COLUMN INSERT` statements must be selected in order to transfer to *MySQL*. Additionally, everything apart from a `CREATE TABLE` or `INSERT` statement must be removed to load the file in *MySQL* because of slight differences in the way sequences are handled and the possibility of different database ownership.

Once the data has been moved into *MySQL* format, the `.sql` file output can be read in by the *MySQL* console using the `source` command and the file name, and the database will be recreated on the server.

The search functionality is implemented in two different ways — typing in to the search box causes suggestions to drop down thanks to a short *Javascript* — `suggestions.js` which requests an SQL query from the server via `search.php` using AJAX[4]. The results returned from the SQL query on each key stroke form the suggestions list. If the user clicks the search button or hits enter while typing, then they are taken to the search results page `fullsearch.php` which takes the search query submitted by the form input element and outputs all results matching the string from the author, paper and keyword tables.

5.3 Procedural generation of HTML files

The PHP code is hosted on the Cealexse server in the directory

`/var/www/html/buntinga/ALEX.`

There are three main PHP files used to display the contents of the database — `authors.php`, `keywords.php`, `papers.php`. Each one of these is intended to be opened with a query string referencing the author, paper or keyword to display. The content of each page is then pulled by PHP code embedded in the page from the database using an SQL query, and the results are returned to the client browser. These could be merged if desired — meaning less work when anyone wants to change the surrounding HTML formatting; however it makes it easier to modify the content of each page without affecting the others in the current layout.

The list pages, `authorlist.php`, `paperlist.php` and `keywordlist.php`, load the full list of authors, papers and keywords respectively again by SQL query. Only the homepage, `homepage.html` and the frequently asked questions (FAQ) page `faq.html` are static HTML. Note that main pages, home page, FAQ page and search page all have their own Cascading Style Sheet (CSS) file at present, however future implementations should consider combining these into one CSS for consistency and maintainability.

6 Analysis of the DST Group collaboration network

Even though there is a lot of room for improvement in the population of the collaboration and citation networks, there is the potential to start exploring these networks for insights

[4] Asynchronous Java Extensible Markup Language (XML)

about DST Group. In this section, we describe a methodology for carrying out this analysis and describe some insights arising from it.

Gephi [2], is a network graph visualisation tool currently in beta development. *Gephi* can be loaded with a list of nodes, for example the authors, and a list of edges, like the collaboration links, into its *data laboratory*. These need to be dumped from *PostgreSQL* using either a *copy to CSV* command from the command line, or using *pgAdmin* to save the result set as a CSV. In order to import into *Gephi*, the following structure of the import data needs to be followed.

- The IDs of the nodes must be included in the node list spreadsheet imported into *Gephi* under the column name `id`. For example importing a list of authors as the nodes, the `author_id` must be included under column name as `id`.
- The two columns of IDs in the edge list defining links between nodes must have column titles `source` and `target` — which one is which is not important. For example a direct export of `author2author` could have column `author1_id` as `source` and column `author2_id` as `target`.
- The edge list should include a column called `type` and every value in that column must be `undirected`. This is easiest to accomplish in *Excel* by modifying the output CSV file so that *Gephi* recognises the links as undirected, otherwise it defaults to assuming directed links.
- Since it is only possible to display an author as existing in one division, if an author had published papers in multiple divisions, currently their division is randomly assigned from the set that they published under. Clearly, other approaches should be considered in future implementations.

Once the data is imported into *Gephi*, there are a number of operations that can be applied to simplify the network to a more practical and comprehensible level for the purpose of analysis.

The first and most important thing to do is to choose a *layout* — a method for producing graphical displays of the network. The general idea of these algorithms is to spatially cluster nodes that have a lot of linkage. There are several layout algorithms included in *Gephi*, with more available via plug-ins, however the two that have been tested in this project are:

- *Futcherman-Reignold*: treats the graph as a mass-spring system, with attraction between connected nodes and repulsion otherwise. Attempts to find a lowest potential energy state for the system. This method is computationally expensive, and while the result is visually appealing it is slow to perform adjustments.
- *Force-Atlas 2*: a layout algorithm that uses similar principles to Futcherman-Reignold; connected nodes are attracted according to the strength of their connection, other nodes are repelled. It makes simplifications allowing it to run significantly faster and on larger datasets.

The `prevent overlap` is an option in *Force-Atlas 2* that makes results a lot easier to interpret by preventing nodes from overlapping. Styling of the graph is also important in order to gain insight — the colour and size of the nodes can be adjusted using the *Gephi* interface.

It is important to note that these graph visualizations are non-unique and are selected through *Gephi*'s user interface. This is done by dragging the nodes across the screen and then allowing the network to settle into a new configuration. The structures we observe below seemed to be consistent under these rearrangements, but further analysis would be required in order to draw definitive conclusions.

6.1 Clustering and divisions

To add the information on authors' divisions to the display, we import a list of `authorIDs` matched to the names of the divisions, and *Gephi* automatically adds the information to the existing author node list.

Colouring by divisions is done using the `partition` tab — click on the refresh arrow then select `division`. The user then manually selects a colour for each division.

6.2 Clustering results

In this subsection we provide a number of visualisations of the DST Group reports collaboration network which are produced using the methods described in the previous subsections. These can be seen in the following figures, which are accompanied by a brief discussion. This descriptive analysis is not the focus of the project and is not meant to be rigorous, but rather illustrative of the type of analysis that becomes possible once the *ALEX* database is populated.

Figure 5 shows the author linkages (collaborations) represented as a network diagram — each node is an author and an edge connecting authors indicates that those authors collaborated at least once. The size and the colour of the nodes indicates the number of papers produced by that author. We note that there are a number of *satellite* authors (and groups of authors) who had a very limited collaboration network. Mainly these are authors of small numbers of papers. Authors with large numbers of papers seem to be more central to the network with some exceptions, such as the author represented by the blue node in the top right of the figure.

Figure 6 is the same as Figure 5, except the authors are now coloured by division (as described previously). As might be expected, authors from the same division appear closer to each other in the graph diagram. This implies that authors in the same division are more likely to collaborate.

Figure 7 makes this observation clearer by limiting the colouring to only two divisions: Maritime Platforms Division (MPD) and Air Vehicles Division (AVD). We still observe that division appears to be a good predictor of clustering, but also that there is significant overlap between the two divisions. The presence of other colours within the clusters may also be due to authors who have changed division.

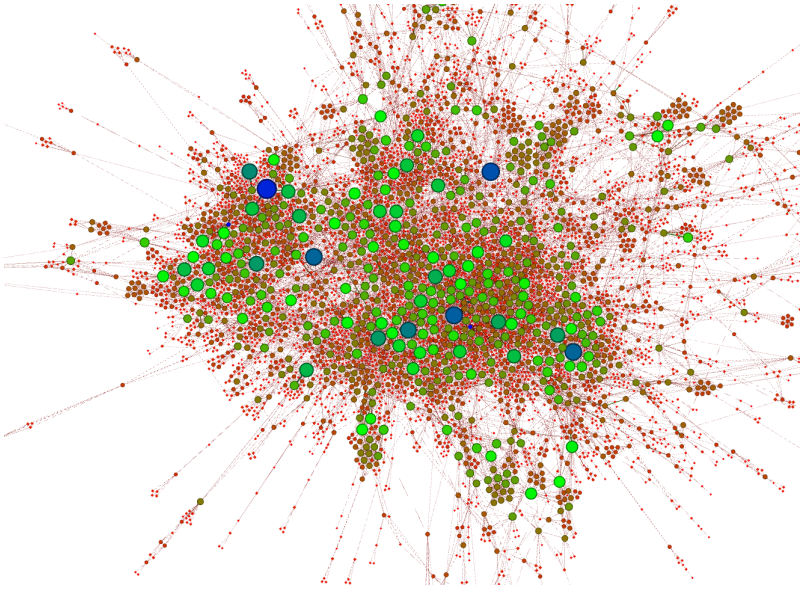

Figure 5: Author linkages represented as a network diagram. The size and colour of the nodes indicates the number of reports produced.

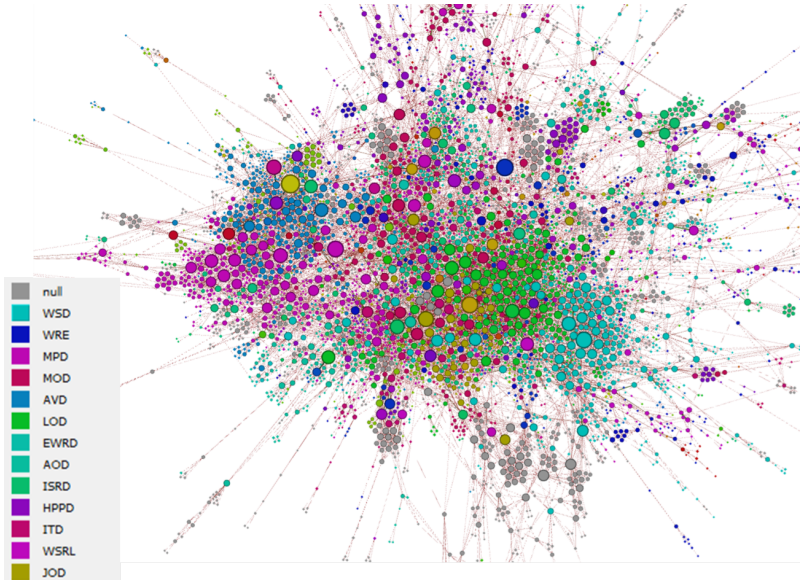

Figure 6: Network of authors coloured by division.

Figure 8 continues this analysis by colouring an additional two divisions: Land Operations Division (LOD) and Weapons Systems Division (WSD). We see that these two divisions are strongly interconnected, but clearly separated from MPD and AVD. Since AVD and MPD are primarily associated with in Melbourne and LOD and WSD with Adelaide, we now investigate the possibility that physical location might be predictor of the degree of interconnectedness of authors.

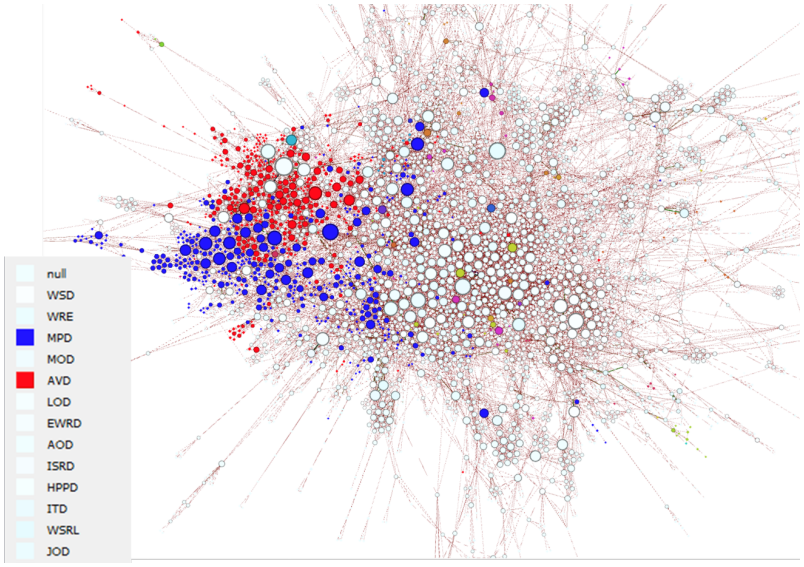

Figure 7: Author network limited to authors who are in MPD or AVD at some point. It provides a clearer insight than the similar Figure 6.

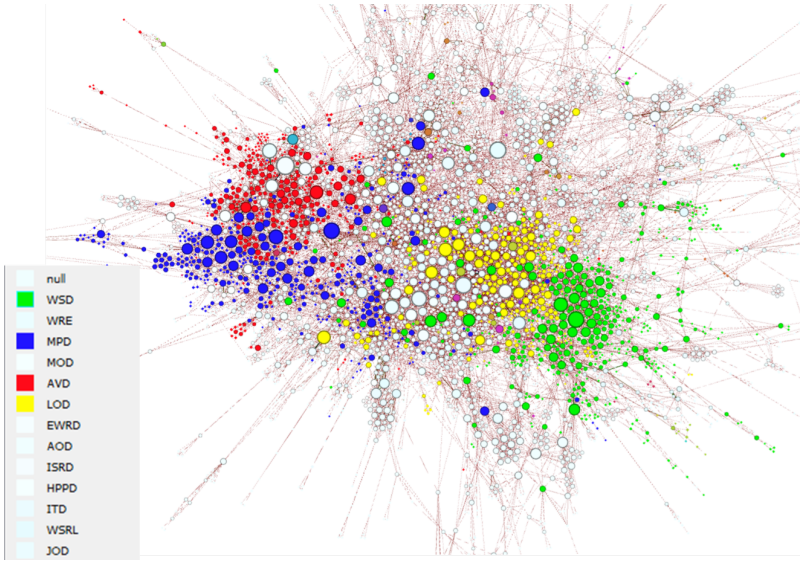

Figure 8: Author network with colours limited to authors who published reports in WSD, MPD, AVD or LOD. Note that WSD and LOD are located at Adelaide and are separated from MPD and AVD which are located in Melbourne.

Figure 9 shows the author network coloured by each division's primary location. The divisions which are mainly based in Melbourne are coloured in red and those in Adelaide are in green. Divisions which are not clearly located in one site — Weapons Research Establishment (WRE) and Maritime Operations Division (MOD) — are coloured black and Joint Operations Division (JOD) is coloured blue to indicate their location in Canberra.

Here we note that physical location appears to be a good predictor of author collaboration.

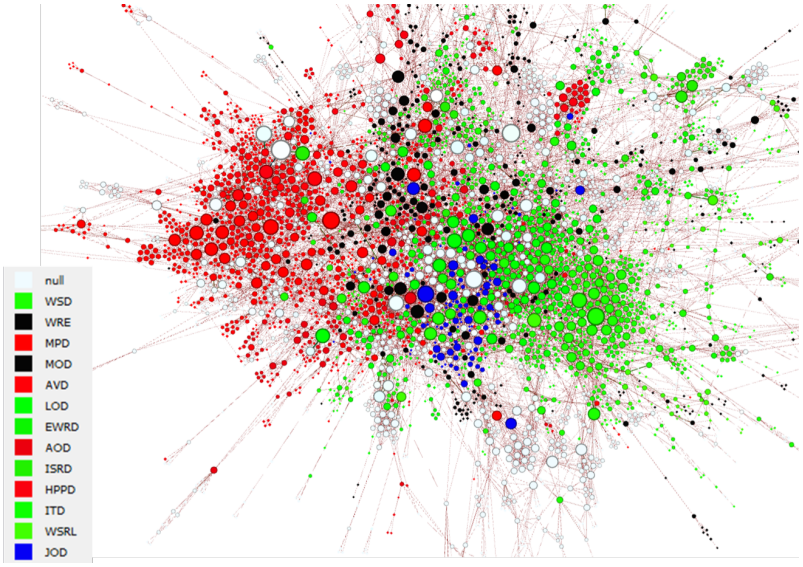

Figure 9: Author network colours by nominal divisional location. This seems to be associated with the degree of interconnectedness.

It is important to also note however that the simple allocation of divisions to sites is not fully accurate — this could potentially mean that the clustering is being over or under represented. Nonetheless, this observation supports the assertion that inter-divisional collaboration may occur within sites more than between sites. Further exploration of the DST Group collaboration and citation networks may provide further and perhaps more concrete insights into the way we work together.

The ease of the preceding analysis, the insights it can provide, and the desirability of exploring further research questions are key benefits in setting up a database such as *ALEX*.

7 Acknowledgements

Particular thanks are due to Vesna Davidovic and Matthew Brightwell from RSG. In addition to supplying the library database they were a valuable source of domain knowledge. Thanks also to Maria Athanassenas and Timothy J. Surendonk for valuable comments and feedback on this report.

8 Future work

This project demonstrates what can be done, quickly, with FOSS tools and access to data. Although no effort is currently being devoted to *ALEX* development, the following questions may be of interest for future study:

- including reports of higher classifications

- better author name fidelity (perhaps through user input as appears on sites such as *ResearchGate*)
- use of *ALEX* as a prototype for a future front-end to access library data after the Objective[5] roll-out
- improved reference finding through improved regular expression usage (or other techniques)
- integration of referencing with external sources (for example, links to articles found in *Google Scholar*)
- improved keyword generation
- generation of publication metrics such as citation rankings and h-scores
- allowing users to access some of the visualisation tools through a web interface
- having the better quality OCR data that is generated in this project exported back into the RSG reports database
- having *ALEX* code placed in a software repository to allow for FOSS-style collaboration by other developers or researchers in DST Group.

We finally note that a number of the ideas and tools discussed in this report are being used in the broader Fleet Data research program.

References

1. Alex Bunting. Auxiliary Library Explorer (ALEX). (internal website), 2014. http://atp-exse.dsto.defence.gov.au/buntinga/ALEX/homepage.html (accessed Aug 2015).
2. Gephi Consortium. Gephi makes graphics handy. (website), 2014. http://gephi.github.io/ (accessed Jan 2014).
3. DSpace development team. Schematic of database file. (website), 2009. http://dspace.org/sites/dspace.org/files/archive/1_5_2Documentation/image/db-schema.gif (accessed Aug 2015).
4. Foolabs. XPdf. (website), 2014. http://www.foolabs.com/xpdf/home.html (accessed Dec 2013).
5. The Apache Software Foundation. Apache POI - the Java API for Microsoft documents. (website), 2014. https://poi.apache.org/ (accessed Aug 2015).
6. Imagemagick. Imagemagick. (website), 2014. http://www.imagemagick.org/script/index.php (accessed Jan 2014).

[5]Objective is a information management system being rolled-out in DST Group at the time of writing and is in broad use in Defence in general.

7. Intel. Whaite paper: Extract, transform and load big data with apache hadoop. (website), 2013. https://software.intel.com/sites/default/files/article/402274/etl-big-data-with-hadoop.pdf (accessed Dec 2015).

8. O. Medelyan. Maui-indexer. (website), 2010. https://code.google.com/p/maui-indexer/ (accessed Jan 2014).

9. Wikipedia Miner. Wikipedia Miner Wiki. (website), 2013. https://github.com/dnmilne/wikipediaminer/wiki (accessed Jan 2014).

10. R. Smith. Tesseract OCR. (website), 2012. http://code.google.com/p/tesseract-ocr/ (accessed Dec 2013).

Appendix A New setup

In this appendix, we outline the steps required to reconstruct the *ALEX* database from scratch. In future implementations, many of these steps could be combined into a single routine.

The following software packages are required: *PostgreSQL*, *TesseractOCR*, *PdfToText*, *Imagemagick*, *Ghostscript*, and 64 bit *Java*.

Once the required software is installed, the following steps are to be carried out:

1. Open the metadata CSV file and save as an *Excel* spreadsheet.

2. Create a new empty database `someDBname` with user `postgres` in *PostgreSQL*.

3. Modify directories in the main `MakeALEX` project as required:

   - database name
   - *DSpace* SQL dump location
   - metadata spreadsheet location
   - asset store location
   - desired text output file location
   - location of *Imagemagick* executable
   - location of *PdfToText* executable

4. *Maui* extraction requires a 64bit operating system, 64 bit *Eclipse*, and 64bit *Java* to access up to 6GB of RAM for the *Maui* keyword extraction process. This must also be set as a virtual machine argument when calling the *Maui* part of the script. In *Eclipse* it can be changed in `Run->Run Configurations->Arguments-VM Arguments -Xmx6000m`.

5. Run the project. The new database should be populated with the *ALEX* information.

6. Dump the database table to a file — use *pgAdmin*, right click on the database, select `backup`. Select a `plain` backup, in the first options tab select data only, in the second options tab select `column inserts` and `insert commands` and finally select only the relevant tables for backup from the `objects` tab:
   - `authorsummary`
   - `author2author`
   - `author2paper`
   - `author2key`
   - `author2key2paper`
   - `librarysummary`
   - `key2paper`
   - `keywordsummary`
   - `paper2paper`
7. Delete the first few lines of the backup file since they are *PostgreSQL* specific and not needed.
8. Log into the *MySQL* database from any user on Cealexse using the command: `mysql -h mccmysql -u <user> -p <password>` .
9. Run `ALEXrefresh.sql` to empty the online database using source command.
10. Run the backup file created above to repopulate the database with new information.

DISTRIBUTION LIST

Auxiliary Library Explorer (ALEX) Development

Alex Bunting, Stephen G. M^{c}Ateer and Justin Beck

AUSTRALIA

Task Sponsor

Chief Defence Scientist	1

S&T Program

Chief - Joint and Operations Analysis Division	1
Research Leader - Maritime Capability Analysis	1
Group Leader - Maritime Mathematical Sciences	1
Task Leader - NAV 07/092 (Dr James Smelt)	1
Science Team Leader - Fleet Data Initiative (Dr Timothy J. Surendonk)	1
Manager Knowledge Management Projects (Ms Vensa Davidovic)	1
Mr Alex Bunting	1
Mr Stephen G. M^{c}Ateer	1
Dr Justin Beck	1

(Total number of **full** copies: 10)

DEFENCE SCIENCE AND TECHNOLOGY GROUP DOCUMENT CONTROL DATA	1. DLM/CAVEAT (OF DOCUMENT)

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Auxiliary Library Explorer (ALEX) Development	Document (U) Title (U) Abstract (U)
4. AUTHORS	5. CORPORATE AUTHOR
Alex Bunting, Stephen G. M^{c}Ateer and Justin Beck	Defence Science and Technology Group 506 Lorimer St, Fishermans Bend, Victoria 3207, Australia

6a. DST GROUP NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DST-Group–TN–1492	016-505	Technical Note	February, 2016

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. No. OF PAGES	12. No. OF REFS
	NAV 07/092	CDS	22	10

13. DST Group Publications Repository	14. RELEASE AUTHORITY
`http://dspace.dsto.defence.gov.au/dspace/`	Chief, Joint and Operations Analysis Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DST GROUP RESEARCH LIBRARY THESAURUS

library science, big data, data analysis, databases, network analysis, information science

19. ABSTRACT

In this report we describe the development of the *Auxiliary Library Explorer* (*ALEX*) which was carried out as part of a DST Group summer vacation scholarship in 2013–14 by Alex Bunting. *ALEX* is a prototype tool motivated by *Google Scholar* which uses the DST Group for-official-use-only (FOUO) internal report database as its input. It provides DST Group researchers with an additional discovery capability and is available (at the time of publication) on the DST Group internal website. We use *ALEX*'s underlying database to explore DST Group's collaboration network and discuss a preliminary result supporting the assertion that inter-divisional collaboration may occur within sites more than between sites.